



UNIVERSITÉ DE
SHERBROOKE

FACULTÉ DES SCIENCES

Rapport final du projet de session Réseaux de neurones - IFT725

Lien du projet : https://github.com/abdellahrami/projet_ift725

<i>Auteur</i>	<i>Matricule</i>
RAMI ABDELLAH	19 143 062
KENDADI MOHAMMED	19 145 177

Présenté à :
pierre marc jodoin

12 avril 2020

Table des matières

1	Introduction	2
2	Description du problème	2
3	Démarche scientifique	2
4	Description des bases de données	6
4.1	cifar10	6
4.2	svhn	7
4.3	Fashion-MNIST	7
5	modèles	8
5.1	CNNVanilla	8
5.2	AlexNet	8
5.3	Resnet	8
6	Analyse des résultats	9
6.1	Résultats obtenu sur BD CIFAR10	9
6.1.1	modèle : CNNVanilla	9
6.1.2	modèle : Alexnet	10
6.1.3	modèle : Resnet	10
6.1.4	analyse et interprétations	11
6.2	Résultats obtenu sur BD Fashion-MNIST	11
6.2.1	modèle : CNNVanilla	11
6.2.2	modèle : Alexnet	12
6.2.3	modèle : Resnet	12
6.2.4	analyse et interprétations	13
6.3	Résultats obtenu sur BD SVHN	13
6.3.1	modèle : CNNVanilla	13
6.3.2	modèle : Alexnet	14
6.3.3	modèle : Resnet	14
6.3.4	analyse et interprétations	15
7	Conclusion et perspectives	15
8	Sources	15

1 Introduction

Dans le cadre du projet de session du cours ift725, on va essayer de traiter un sujet qui aborde un thème parmi ceux vus en classe de façon approfondie dans une équipe de deux personnes. Dans ce projet il nous a été demandé d'utiliser la bibliothèque pytorch tout en utilisant un outil de gestion de projet comme git ou bitbucket.

2 Description du problème

Ces jours-ci, nous sommes exposés à une très grande quantité de données non étiquetées provenant d'Internet ou d'une autre source telle que le monde universitaire ou le monde des affaires. Étant donné que les données non étiquetées sont relativement faciles à acquérir et coûteuses à étiqueter, les entreprises emploient généralement un expert ou plusieurs employés dont le but est d'étiqueter les données. Considérons la situation suivante, une entreprise médicale basée sur les données a beaucoup d'exams d'IRM et ils doivent employer un expert qui les aidera à interpréter ces exams. L'entreprise dispose de ressources limitées et ne peut pas interpréter ou étiqueter toutes ses données ; c'est le point où ils décident d'utiliser l'apprentissage actif (AL). La promesse de l'AL est que étant donné une base de données non étiquetées, on va pouvoir sélectionner un sous-ensemble qui a le plus de valeur en termes d'apprentissage, et ainsi on pourra entraîner le mieux possible notre modèle avec le moins possible de données étiquetées. Faire de l'apprentissage avec de l'AL est considérée comme une méthode semi-supervisée, car elle utilise un ensemble de données étiquetées et non étiquetées, et donc le rôle de l'AL et de décider parmi les données non étiquetées ceux qui vont permettre la plus rapide croissance possible de la performance du modèle en se basant, ou pas, sur un modèle qui a utilisé que les données initialement étiquetées pour comprendre le problème.

3 Démarche scientifique

Vu que la notion d'apprentissage actif (AL) se base principalement sur les méthodes d'échantillonnage des données(sampling) on a décidé de nous y prendre en appliquant différentes méthodes trouvés dans nos recherches.

Toute fois on a divisé ces méthodes selon deux approches :

1. méthodes naïves :

On a nommé cette méthode par "naive" car c'est la méthode la plus simple. Cette méthode consiste à entraîner le modèle au départ sur la partie des données déjà étiquetées, puis faire grâce à ce modèle des prédictions d'étiquettes de toutes les données non étiquetées en spécifiant la certitude de chaque prédictions. Par la suite en se basant sur cette certitude on va spécifier les données sur lesquelles le modèle était le moins certain dans sa prédition pour les étiquetées par l'expert, et refaire l'apprentissage en utilisant les données anciennement étiquetées et les nouvelles données étiquetées. Ensuite on répète le même processus en utilisant le nouveau modèle entraîné.

Dans cette méthode on a utilisé trois variantes. La différence entre chacune d'elle est dans la manière dont on calcul la certitude du modèle sur une donnée :

- **La variante 'maximum'** : Dans cette variante on prend la sortie du modèle sur l'entrée de chaque données non étiquetées, on applique un softmax pour obtenir la prédiction en forme de probabilités sur chaque classe des classes possible du problème. Puis on prend pour chaque donnée la plus grande probabilité, qui représente donc la classe prédictive, et enfin les données à étiquetées sont ceux qui ont cette valeur de probabilité la plus petite parmi toutes les données non étiquetées. Car plus cette valeur est petite plus la prédiction du modèle est moins sûre de sa prédiction.
- **La variante 'difference'** : Dans cette variante, au lieu de prendre pour chaque données que la probabilité maximale sur sa prédiction, on prend la différence entre cette probabilité maximale et celle qui suit (la deuxième classe), et puis, de même, les prochaines données à étiquetées sont ceux qui ont cette valeur la plus petite. Car plus cette valeur est petite, plus le modèle n'est pas certain si l'instance appartient à la classe avec la probabilité maximale ou celle avec la deuxième probabilité maximale.
- **la variante 'entropy'** : Dans cette variante, on prend pour chaque données toutes les probabilités en sortie du softmax et on calcule l'entropie de cette liste de probabilités

$$H(x) = -p(x).\log(p(x))$$

en faisant le somme sur toutes les classes. Puis les données à étiquetées sont ceux qui ont cette valeur d'entropie la plus grande.

2. méthode basé sur le clustering :

Dans cette méthode, à chaque itération, pour déterminer le meilleur sous ensemble de données à étiquetées on va pas utiliser les données déjà étiquetées, au lieu on va faire un clustering sur toutes les données non étiquetées, où on va spécifier le nombre de clusters à trouver égale au nombre de classes réelles.

Puis après le clustering on prend le même nombre des données de chacun des clusters, et les données à prendre seront ceux les plus proches du centre du leur cluster.

Pour faire le clustering on a utilisé un réseau de neurones de la forme "**Encoder-Decoder**" basé sur la structure de l'UNet mais avec moins de couches intermédiaire pour accélérer la tâche, puis au milieu du réseau on transforme la "feature map" de la plus petite taille en vecteur linéaire, on donne ça à des couches "Fully connected layers" qui réduisent la taille de ce vecteur linéaire jusqu'à une taille **N** qu'on choisit, puis on refait le contraire pour reconstruire la taille de la feature map, et à la fin du réseau on obtient une sortie similaire à l'entrée. Ceci permet d'entraîner ce modèle avec les données non étiquetées en donnant en entrée et en sortie (label) la même donnée.

Après entraînement, le réseau apprend donc à réduire les données en des vecteurs de taille **N** qu'on choisit comme un nombre assez petit. Donc pour chaque donnée non étiquetées on la roule sur ce modèle et on prend la sortie de la première moitié du réseau (la partie Encoder).

Avec cette transformation de la dimensionnalité des données, on peut maintenant facilement utiliser un algorithme classique. On a utilisé donc "**Kmeans**" en spécifiant le **K** égale au nombre de classes qui existe réellement.

En spécifiant le N dans le réseau "**Encoder**" à $N = 2$, on a tracé un graphique de quelque données et on a utilisée les étiquettes réel pour voir si les données sont assez séparées, et ceci sur les trois bases de données avec lesquelles on va appliquer l'AL. On a obtenu ce qui suit.

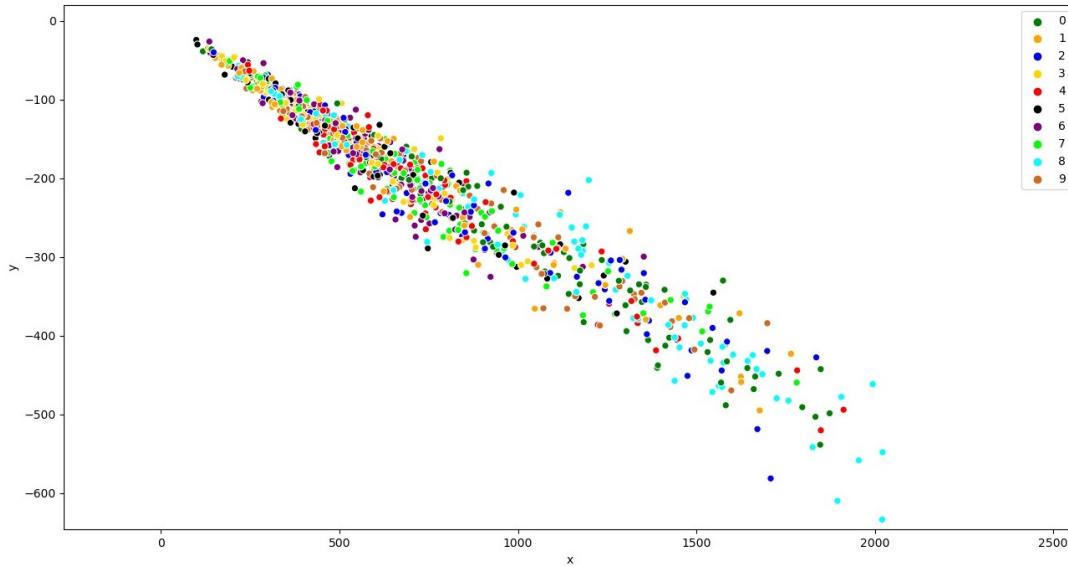


FIGURE 1 – Réduction de la dimensionnalité à 2 de la BD cifar10

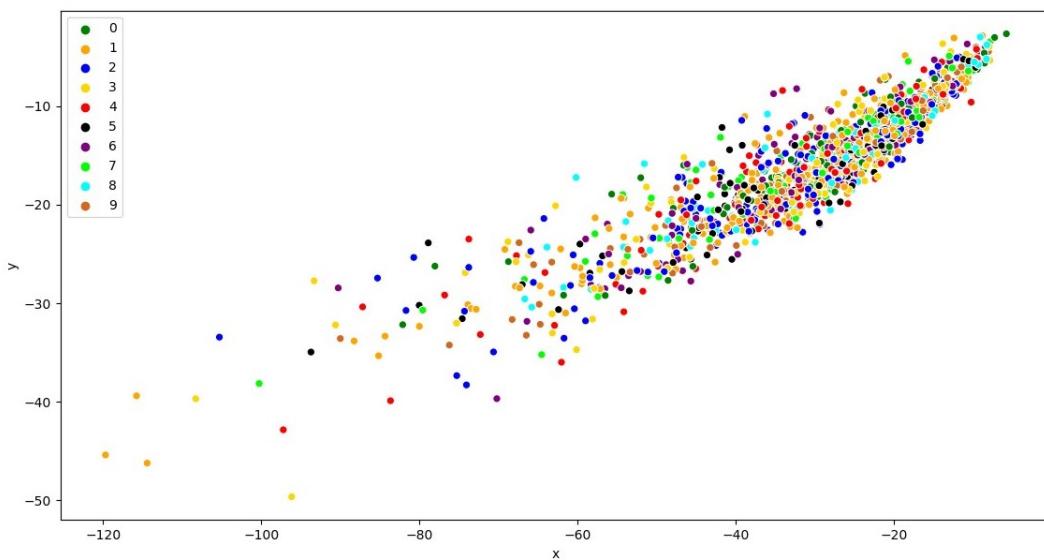


FIGURE 2 – Réduction de la dimensionnalité à 2 de la BD svhn

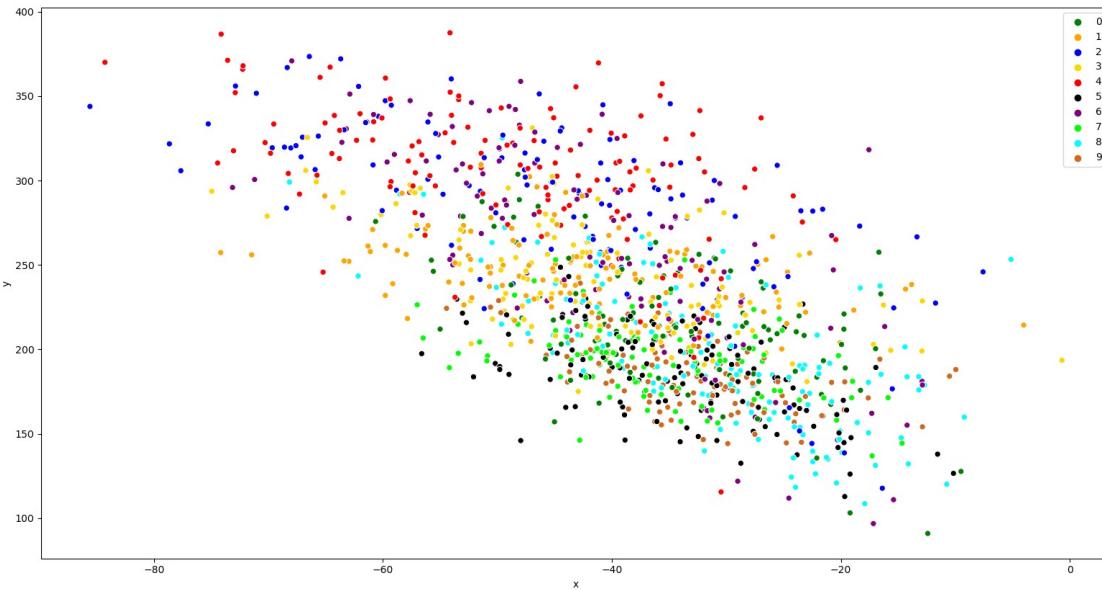


FIGURE 3 – Réduction de la dimensionnalité à 2 de la BD Fashion-MNIST

En analyse de ces figures, on constate que pour les BD **cifar10** et **svhn** les données sont pas du tout séparées. Mais pour la BD **Fashion-MNIST** les données sont un peu séparées mais avec des superpositions entre quelques classes, comme entre la classes 2 et 4, la classes 3 et 1, et puis entre les classes 0, 5, 7, et 8. Sur la BD **Fashion-MNIST** on trouve par exemple que la classe 2 représente les "pullover" et la classes 4 représente les "coat" et qui sont assez similaire surtout sur des images en niveau de gris.



FIGURE 4 – Exemple de la classe 2 de la BD Fashion-MNIST



FIGURE 5 – Exemple de la classe 4 de la BD Fashion-MNIST

Finalement pour pouvoir savoir si c'est modèle sont bon ou pas, c'est à dire s'ils apportent un plus ou pas, on a aussi implémenté une méthode qui choisis les données à étiquetées aléatoirement.

Les méthodes citées ci-dessus vont être expérimentées avec différentes bases de données, on a choisis les trois BD suivantes : **cifar10**, **svhn**, et **fashion-mnist** qui sont toutes des bases de données d'images. on verra les spécificités de chacune de ces bases de données par la suite.

L'application du AL sur ces 3 bases de données va être faite en utilisant différents modèles familiers pour nous, qu'on a vu en classe et manipulé lors du tp3, en particulier on a choisis les trois modèles suivants :

- **CNNvanilla**
- **AlexNet**
- **ResNet**

Le processus qu'on va utiliser pour appliquer chacune des méthodes de l'AL sera d'utiliser la méthode pour spécifier à chaque itération quel est la prochaine 10% de la BD qu'on doit étiquetée, puis entraîner le modèle de nouveau sur toutes les données déjà choisis dans les itérations précédentes et les nouvelles données choisis dans cette itération. Puis dans chaque itération on calcule la justesse du modèle sur l'ensemble de test qui reste le même inchangé depuis le début du processus.

Donc à la fin on obtient 10 valeurs de justesse sur l'ensemble de test, en utilisant respectivement 10% puis 20% puis ... jusqu'à 100% des données de l'entraînement de la BD choisis. Ces valeurs vont être tracer sur un graphique qu'on présente dans la partie des résultats pour comparer entre les méthodes.

4 Description des bases de données

4.1 cifar10

contient 60000 images dont 10k de test et 50k données d'entraînement, la taille des images dans cifar10 est $3 \times 32 \times 32$, le 3 signifie que les images sont composés de 3 couleurs principales RGB .

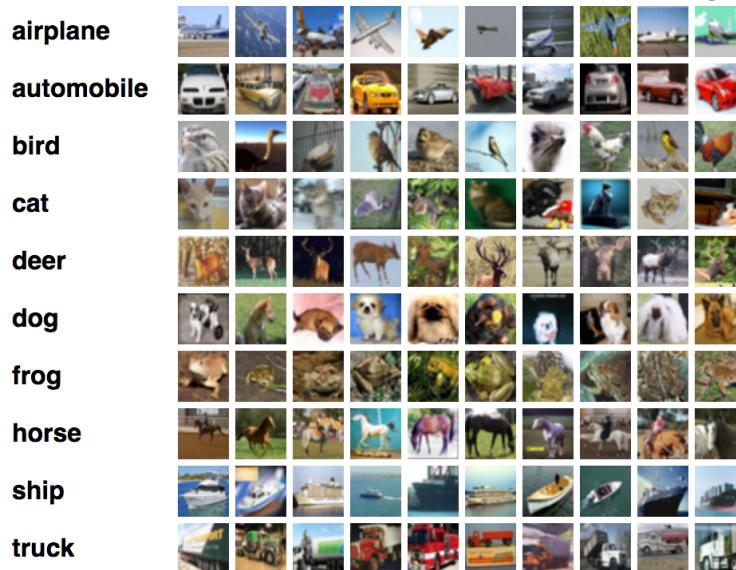


FIGURE 6 – exemple des données de cifar10

4.2 svhn

c'est une base de données de nombres, contient plus de 600k images de taille 3*32*32 à 10 différentes classes, tel que chaque classe correspond à un chiffre de 0 à 9. (le nombre 1 a une étiquette de 1, 2 une étiquette 2 ... et 0 une étiquette 10)



FIGURE 7 – exemple des données de svhn

4.3 Fashion-MNIST

taille de l'image : 1*28*28 Celle ci est en noir et blanc et la taille d'images est aussi différente par rapport aux Deux autres bases de données. Elle contient les images de vêtements, qu'on va essayer de classifier selon 10 classes :

0. **T-shirt / haut**
1. **pantalon**
2. **Pull**
3. **Robe**
4. **Manteau**
5. **Sandale**
6. **Chemise**
7. **Sneaker**
8. **sachet**
9. **Bottines**

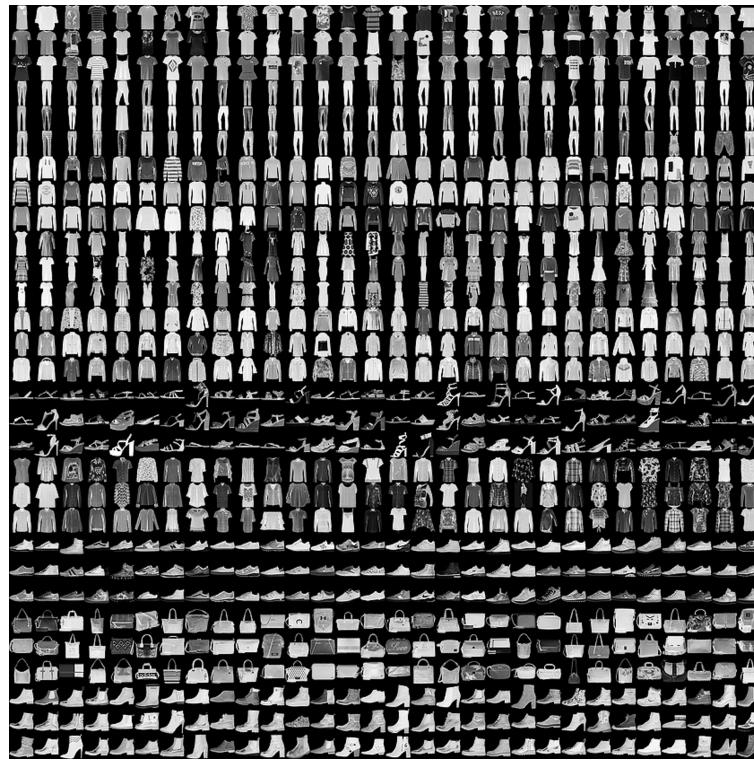


FIGURE 8 – exemple des données de fashion-mnist

5 modèles

5.1 CNNVanilla

C'est le modèle le plus simple, il se compose principalement des couches convolutives suivies de couches pleinement connectées.

5.2 AlexNet

Ressemble beaucoup à cnn vanilla, en principe ils ont la même structure, sauf qu'on a ajouté un pooling par moyenne à l'entrée de la couche pleinement connectée et ce pour réduire le nombre de paramètres.

5.3 Resnet

C'est le modèle le plus profond, sa structure est composée de couches convolutives suivies de 4 couches résiduelles (resnet18) ensuite une couche pleinement connectée avant de faire la prédiction. Donc on s'attend à ce que ce modèle soit le plus performant.

Remarque : Pour les modèles ci-haut on a principalement utilisé le code fourni dans le tp3.

6 Analyse des résultats

6.1 Résultats obtenu sur BD CIFAR10

6.1.1 modèle : CNNVanilla

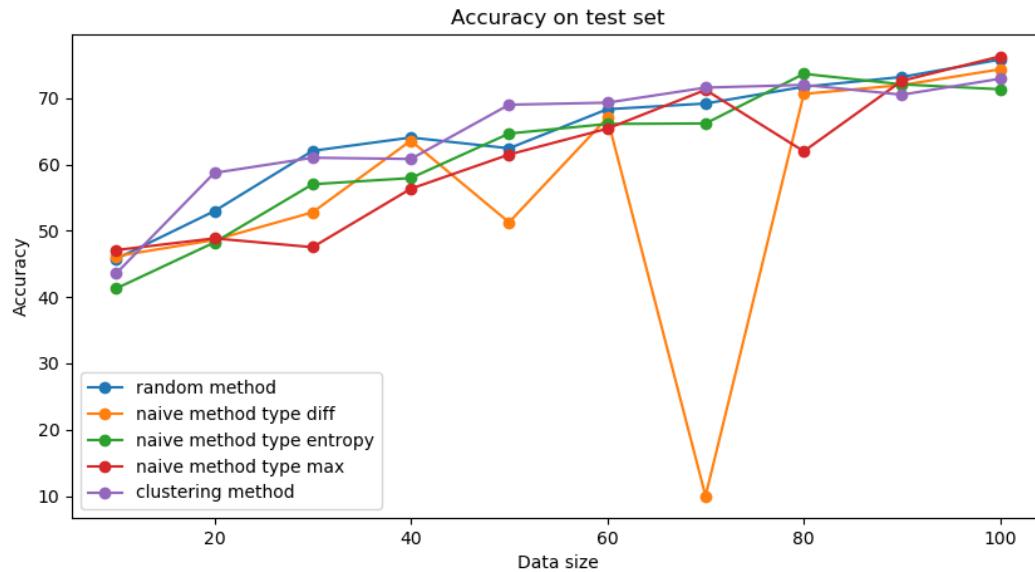


FIGURE 9 – Résultats sur Cifar10 en utilisant CnnVanilla

6.1.2 modèle : Alexnet

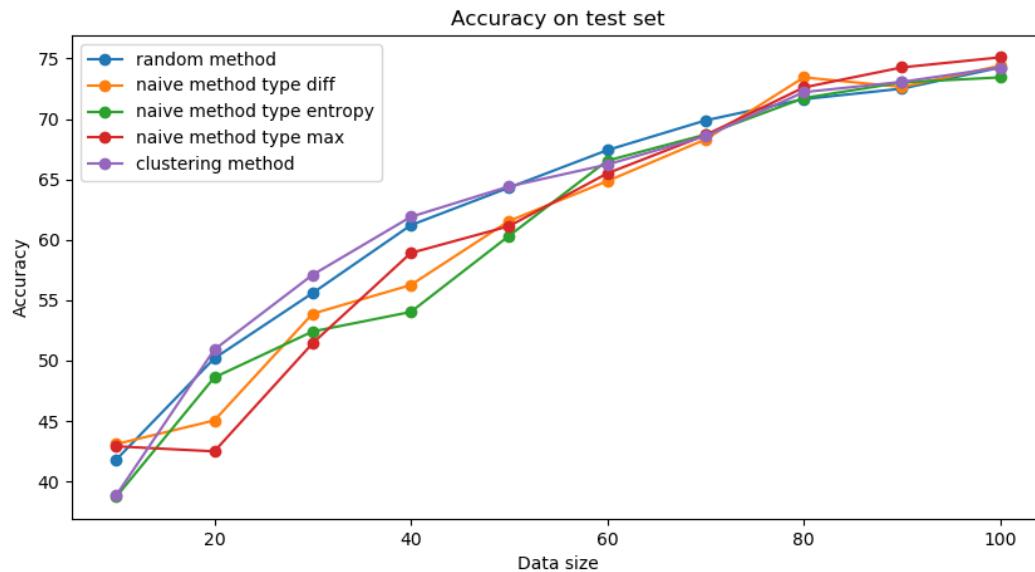


FIGURE 10 – Résultats sur Cifar10 en utilisant AlexNet

6.1.3 modèle : Resnet

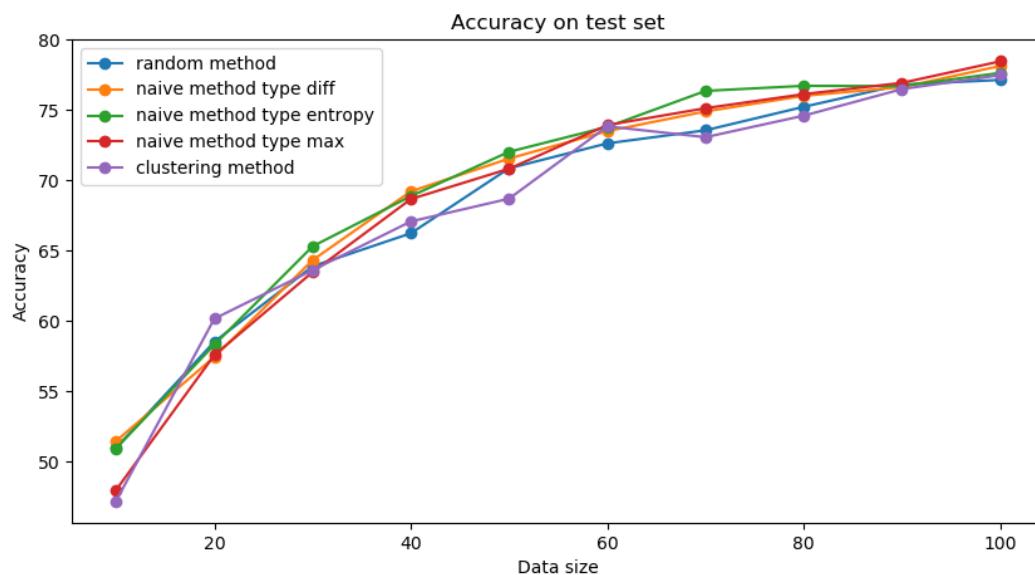


FIGURE 11 – Résultats sur Cifar10 en utilisant ResNet

6.1.4 analyse et interprétations

En ce qui concerne la base de données cifar10, on remarque que pour le modèle **ResNet** les 3 variantes de la méthode naïves tout comme la méthode basée sur le clustering se débrouille d'une façon un peu meilleur que la méthode aléatoire, surtout dans les itérations 40% et 70%. Alors que pour les modèles CNNVanilla et AlexNet, seule la méthode basée sur le clustering permet d'avoir des résultats un peu meilleur que celle de l'aléatoire dans les premières itérations, alors que la méthode naïve donne des résultats pires que la méthode aléatoire.

Ceci est peut être dû à la récupération des données basée simplement sur la probabilité d'appartenance aux différentes classes dans les méthodes naïves, chose qui peut parfois causer que les données ne sont pas bien réparties sur toutes les classes réelles, en plus de la possibilité qu'il contiennent des données aberrantes qui ne vont pas aider le modèle à mieux généraliser. Alors que pour la méthode basé sur le clustering, on choisit les données les plus représentatives de chaque cluster (les plus proches du centre) avec une répartition égale sur tout les clusters.

6.2 Résultats obtenu sur BD Fashion-MNIST

6.2.1 modèle : CNNVanilla

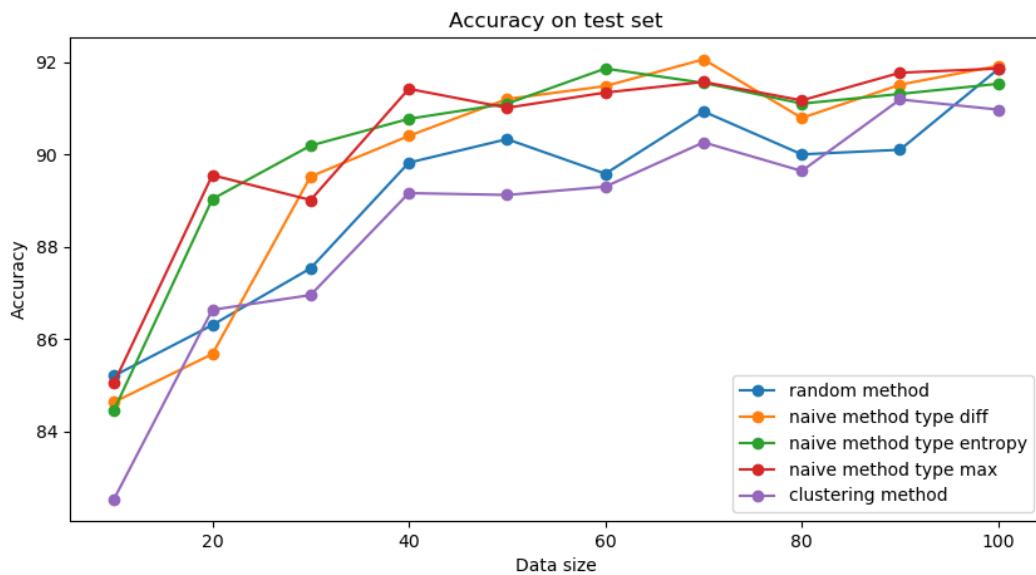


FIGURE 12 – Résultats sur Fashion-MNIST en utilisant CnnVanilla

6.2.2 modèle : Alexnet

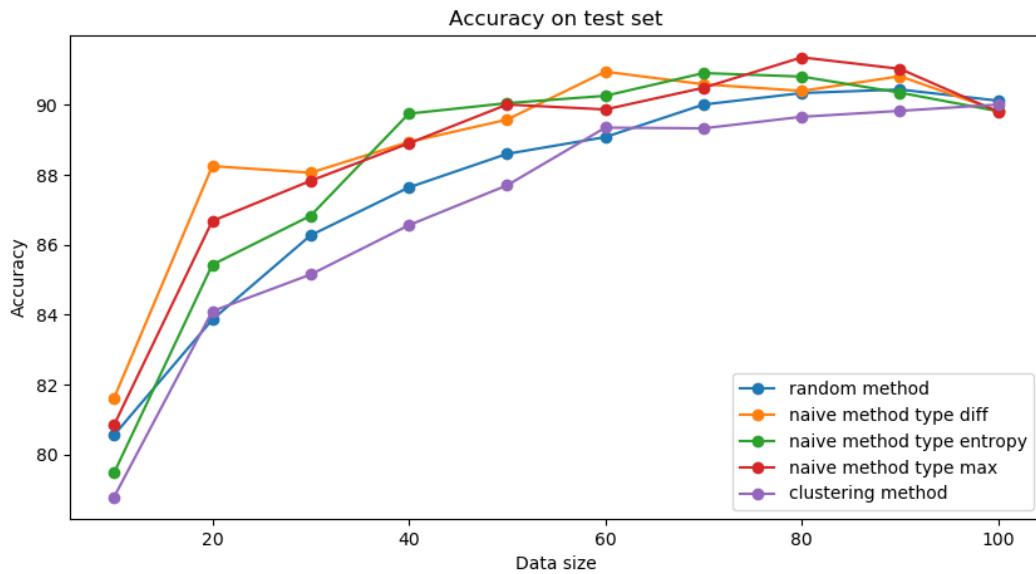


FIGURE 13 – Résultats sur Fashion-MNIST en utilisant AlexNet

6.2.3 modèle : Resnet

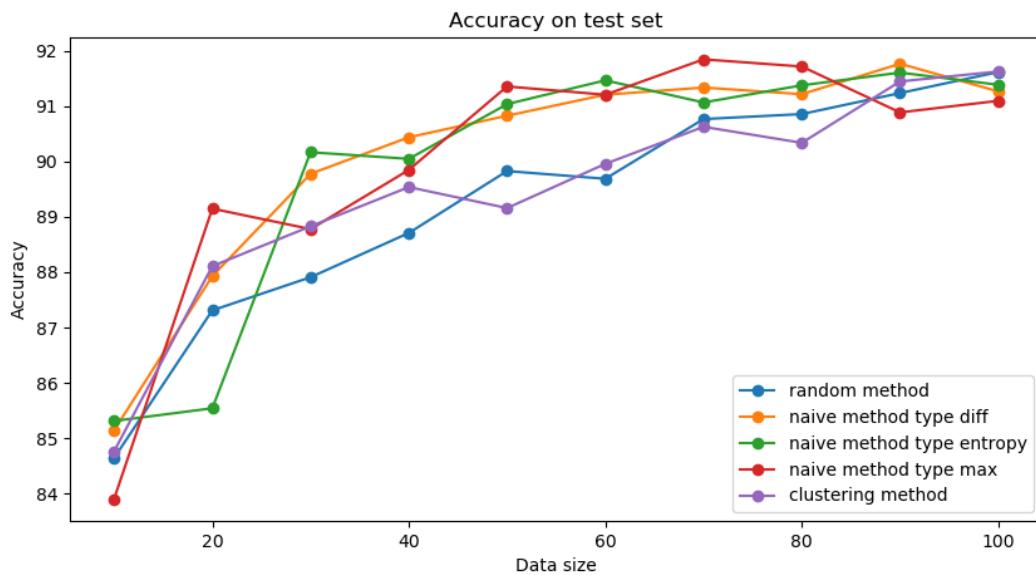


FIGURE 14 – Résultats sur Fashion-MNIST en utilisant ResNet

6.2.4 analyse et interprétations

Pour Fashion-mnist on trouve que l'apprentissage actif est très performant avec les méthodes naïves qui se démarquent de l'aleatoire tout au long des itérations. On note que la méthode naïve type max atteint une justesse vraiment proche de la justesse atteinte si on utilise 100% de nos données dès 40% sur le modèle CnnVanilla. La même chose s'applique pour le modèle AlexNet et ResNet.

On remarque aussi que la méthode basée sur le clustering généralement se débrouille un peu comme (ou parfois pire que) la méthode aléatoire dans cette base de données. On pourra expliquer ça par le fait que la méthode de clustering (kmeans) qui a été utilisée n'était pas très performante et donc les différents cluster retournés ne représente pas les vraies classes de la base de données.

6.3 Résultats obtenu sur BD SVHN

6.3.1 modèle : CNNVanilla

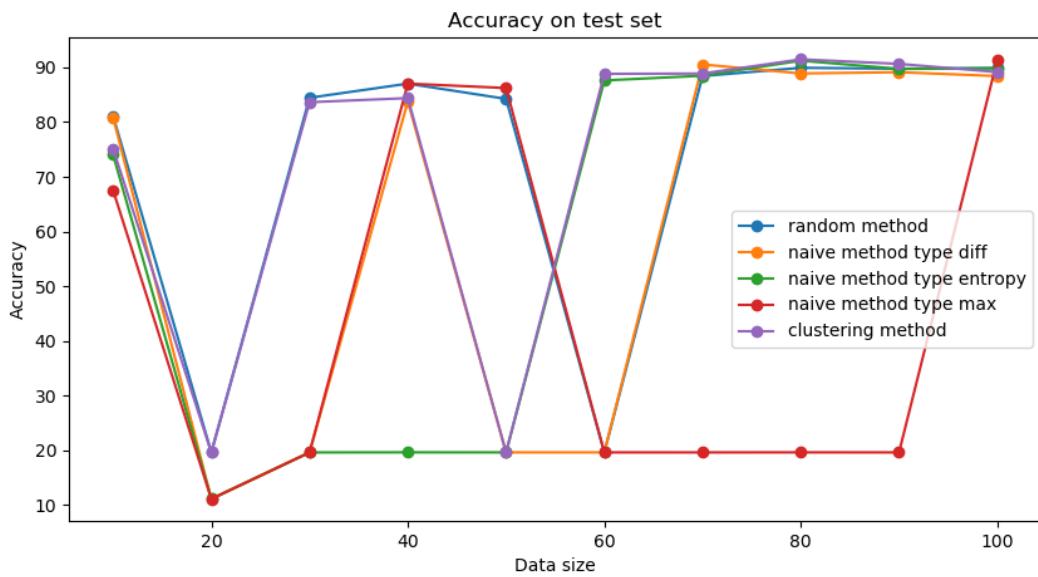


FIGURE 15 – Résultats sur svhn en utilisant CnnVanilla

6.3.2 modèle : Alexnet

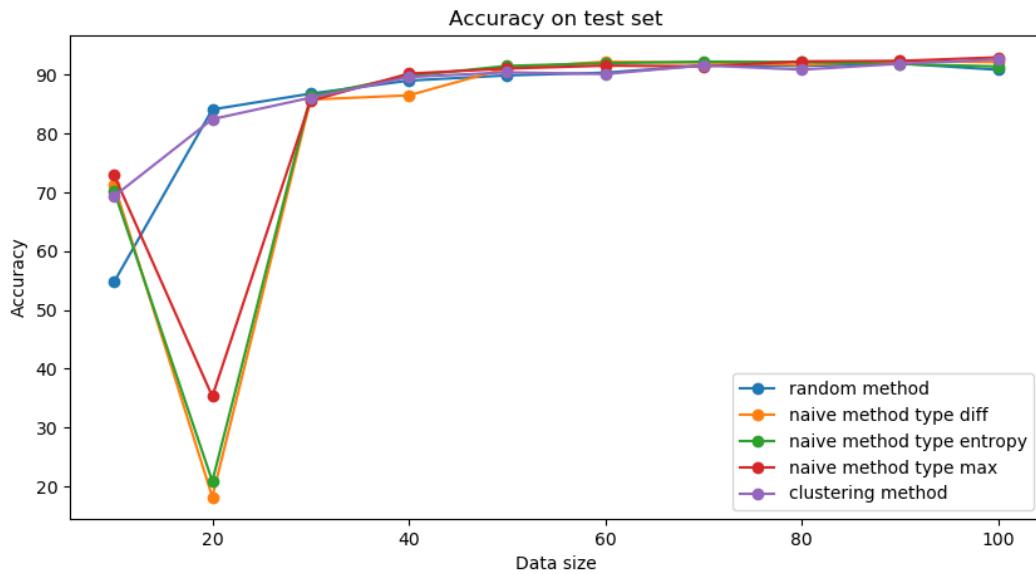


FIGURE 16 – Résultats sur svhn en utilisant AlexNet

6.3.3 modèle : Resnet

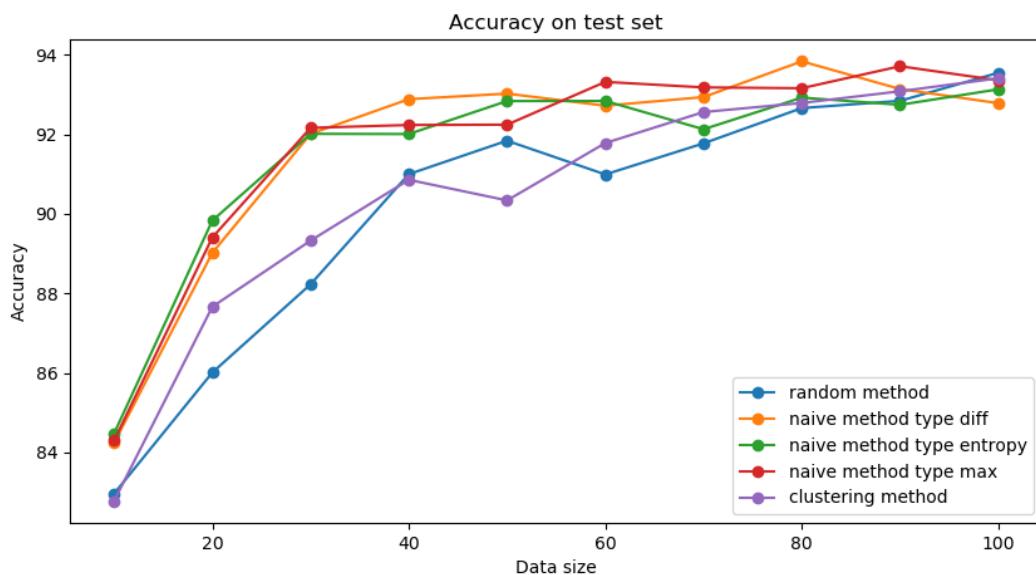


FIGURE 17 – Résultats sur svhn en utilisant ResNet

6.3.4 analyse et interprétations

pour CNNVanilla, on remarque que les résultats sont désordonnées, ils nous permet pas de tirer de grands informations, sauf que pour la méthode naïve entropy et la méthode basé sur clustering qui eux atteint le maximum à partir de 60% des données tout comme la méthode aléatoire. on conclut que CNNVanilla performe de façon moins bonne par rapport aux deux autres modèles ci dessous.

Pour AlexNet on voit qu'à l'itération des 20% des données les variantes de la méthode naïve donne un très mauvais résultats, cela peut encore revenir au fait que les données choisies par cette méthodes représente juste quelques classes et pas d'autres ce qui affecte négativement la capacité du modèle à généraliser. Puis après les 30% on voit que les 5 méthodes, celle aléatoire y comprise, donne tous presque le même résultats, et qui est un résultat dès 30% des données très proche du résultats final avec 100% des données. Donc juste avec 30% des données AlexNet est capable de comprendre la structure du problème, ainsi, avec ou sans AL, on obtient une justesse suffisante.

Finalement pour ResNet, on a obtenu d'assez bon résultats, où on constate les variantes de la méthode naïve donne des résultats bien meilleur que la méthode aléatoire dès les premières itérations, alors que la méthodes basé sur le clustering dépasse un peu la méthode aléatoire au début mais finit par donné des résultats similaires ou même pire sur les itérations suivantes.

Ceci nous informe aussi que le modèle ResNet, grâce a sa forte capacité, est moins vulnérable au changement qui ont affecté négativement les autres modèles dans les itérations des 30% des données.

7 Conclusion et perspectives

En conclusion on peut dire que ces méthodes n'ont pas été un succès absolu, vu qu'ils performent vraiment bien sur quelques bases de données et quelques modèles mais moins bien sur d'autres. Ceci peut revenir au fait du mauvais choix de quelque hyper-paramètres comme le nombres d'epochs sur lesquels on entraîne notre modèles dans chaque itération. Ainsi, l'une des perspectives qu'on aurait bien voulu appliquer si on avait plus de temps est de faire une validation croisée sur l'ensemble des hyper-paramètres(comme le nombre des couches des différents réseaux), aussi de tester d'autres méthodes de l'active learning qui se basent sur des idées plus complexes.

8 Sources

1. Méthodes d'active learning : [link](#)
2. Méthode basée sur le clustering : [link](#)
3. Les modèles utilisés : [link](#)